NXPU: A Heterogeneous Neurosymbolic Processor Architecture

with Learned Dynamic Orchestration and Grounded Reasoning Intelligence

Zachary Kleckner

Dyber, Inc.

zkleckner@dyber.org

Abstract

We present the NXPU, a heterogeneous processor architecture integrating three specialized engines for neurosymbolic computation: a Neural Mesh (NM) for spiking neural network processing with online STDP learning, a Symbolic Logic Unit (SLU) with content-addressable memory for hardware-accelerated Datalog inference, and a Causal Simulation Engine (CSE) for probabilistic causal reasoning with native do-calculus support. An Orchestrator Core (OC) learns to dynamically route sub-problems to the appropriate engine using a compact policy trained via REINFORCE, exportable to a 512-entry hardware lookup table. We evaluate the architecture through NXSim, a cycle-approximate software simulator, on standard benchmarks including the Sachs protein signaling network for causal discovery, multi-hop knowledge graph reasoning over 11,904 triples, and one-shot pattern classification with spiking networks.

Results demonstrate: (1) interventional causal discovery achieves F1=0.786 on the Sachs benchmark, beating the PC algorithm (0.400); (2) persistent reward-modulated STDP enables 100% one-shot classification from random initial connectivity with single-exposure learning (no gradient-based training); (3) the NXRI reasoning layer achieves 87% compositional reasoning accuracy across 12 difficulty levels and generalizes hypothesis testing to 4 domains (medical, coding, legal, engineering) with correct Bayesian evidence updating; (4) learned orchestration matches oracle routing performance at 3x lower energy. Based on published per-operation energy costs from Intel Loihi and 7nm SRAM literature, we estimate potential energy improvements of several orders of magnitude over GPU baselines for symbolic and causal workloads, though these projections require hardware validation. All results are achieved without gradient-based training. The NM requires single example exposures with reward signals; the SLU requires a loaded knowledge base; the CSE requires interventional access to the environment. None require large training datasets or GPU compute. The system includes NXRI (NX Reasoning Intelligence), a novel reasoning layer that provides epistemic status tagging, inductive rule discovery, and gap detection — an anti-hallucination mechanism absent from current large language models.

Keywords: *neurosymbolic computing, heterogeneous processor, spiking neural networks, STDP, causal inference, do-calculus, Datalog, content-addressable memory, learned orchestration, FPGA, neuromorphic, low-data reasoning, epistemic status*

I. Introduction

A. The Problem with Single-Paradigm AI Hardware

Current AI hardware is optimized for a single computation paradigm: dense matrix multiplication for neural network training and inference. While GPUs and TPUs excel at this workload, three critical AI capabilities remain without dedicated hardware support:

- Symbolic reasoning: logical inference over knowledge graphs requires content-addressable memory (CAM) and unification, not matrix multiply
- Causal inference: counterfactual reasoning requires graph surgery and probabilistic sampling, not gradient descent
- Online learning: adapting to new patterns from single exposures requires spike-timing-dependent plasticity (STDP), not backpropagation over large datasets

These limitations manifest in practical failures: LLMs hallucinate because they lack grounded reasoning; recommendation systems cannot answer "what would have happened if we had changed X" because they lack causal inference; and deployed models require expensive retraining to incorporate new information because they lack online learning.

B. The NXPU Approach

The NXPU integrates three heterogeneous engines on a single die, each optimized for a different computation paradigm, connected by a Unified Memory Fabric (UMF) with tagged provenance and coordinated by a learned meta-controller. The programmer writes in NXLang, a domain-specific language that compiles through an SSA-form intermediate representation (NXIR) to engine-specific backend code.

C. Contributions

- The first processor architecture integrating spiking neural networks, symbolic Datalog inference, and causal do-calculus on a single chip
- A learned Orchestrator that dynamically routes sub-problems to engines, matching oracle performance at 3x lower energy than parallel execution
- Persistent reward-modulated STDP with Elastic Weight Consolidation that achieves 100% one-shot learning from random connectivity without catastrophic forgetting
- NXRI (NX Reasoning Intelligence), a reasoning layer providing gap detection, abductive hypothesis generation, inductive rule discovery from data, and calibrated epistemic status for every output
- Comprehensive evaluation on standard benchmarks (Sachs et al. [1], ARC-AGI [6]) with real baselines (PC algorithm, Z3 solver, published Loihi numbers)

II. Architecture

The NXPU comprises four functional units sharing a tagged memory fabric. Each unit is optimized for a distinct computation paradigm. This section describes each unit, its computational model, and Phase 0 simulation results.

A. Neural Mesh (NM)

The Neural Mesh implements a spiking neural network with Leaky Integrate-and-Fire (LIF) neurons and configurable STDP learning rules. Unlike conventional neural accelerators that require offline training on large datasets, the NM learns online from single exposures through spike-timing-dependent plasticity.

Key innovations in the Phase 0 implementation:

- Persistent network state — synaptic weights maintained across pattern presentations via a long-lived Brian2 Network object, avoiding the state reset that prevents learning in prior neuromorphic simulators
- Reward-modulated STDP (R-STDP) with eligibility traces — an eligibility variable e accumulates spike-timing correlations, but weight updates only occur when an external reward signal is delivered: $\Delta w = \eta \cdot R(t) \cdot e(t)$
- Elastic Weight Consolidation (EWC) [8] — Fisher information matrix tracks weight importance; previously important weights resist modification during new learning, preventing catastrophic forgetting
- Homeostatic plasticity via adaptive thresholds — neurons that fire excessively raise their threshold ($d\theta/dt = (\theta\_target - \theta) / \tau\_h$), preventing runaway excitation while maintaining network responsiveness

Phase 0 results: 100% accuracy on 4-class and 8-class classification tasks from random initial connectivity (no architectural bias), across 5 seeds (42, 137, 256, 999, 7777), scaling to 1024 neurons and 64 input channels. The classification tasks use synthetic spike-rate patterns where each class activates a distinct, non-overlapping group of input channels (group_size = n_channels / n_classes) at high rates (3.0–5.0 Hz) against a low baseline (0.1–0.4 Hz). Test inputs add uniform noise (±0.3). These patterns have high class separability by design; the result demonstrates that persistent R-STDP can self-organize selectivity from random wiring, not that the task is difficult by machine learning standards. Average neuron activation sparsity: ~50%.

B. Symbolic Logic Unit (SLU)

The SLU implements a hardware-accelerated Datalog inference engine. In the ASIC target, triple lookup is performed via content-addressable memory (CAM) for O(1) pattern matching. The Phase 0 simulation uses Python dictionaries with hash-based indexing.

- Semi-naive bottom-up fixed-point evaluation with stratified negation-as-failure
- Incremental maintenance — when new facts arrive, only rules whose body predicates are affected are re-evaluated, enabling real-time streaming fact ingestion
- Hot-reloadable knowledge bases without recompilation or restart

Phase 0 results: 0.31ms average query latency on 11,904 triples (4,085 base + 7,819 derived) across 150 queries including 5+ hop transitive closure. 7x faster than Z3 SMT solver [5] on equivalent Datalog-style queries.

C. Causal Simulation Engine (CSE)

The CSE implements Pearl's do-calculus [4] for causal reasoning. Given a causal directed acyclic graph (DAG) with conditional probability tables, the CSE supports observational queries, interventional queries via graph surgery (removing incoming edges to the intervention target and fixing its value), and counterfactual evaluation.

Active causal discovery uses information-gain-guided intervention selection: for large graphs (>6 nodes), the system prioritizes interventions that maximize expected entropy reduction across all variables, reducing the total number of interventions needed by approximately 40% compared to exhaustive pairwise testing.

Phase 0 results: F1=0.786 on the Sachs protein signaling network [1] (11 nodes, 17 edges), with 100% precision and 64.7% recall. This exceeds the PC algorithm (F1=0.400), GES (0.400), NOTEARS [7] (0.450), and DAG-GNN (0.500). The CSE wins on all three standard benchmarks tested: Asia (8 nodes), Sachs (11 nodes), and ALARM (37 nodes).

D. Orchestrator Core (OC)

The Orchestrator implements a learned routing policy trained via REINFORCE policy gradient. An 8-dimensional task feature vector (task type, input size, data modality flags, complexity estimate, prior engine confidence) maps to 4 actions (route to NM, SLU, CSE, or run all in parallel). The softmax policy is a linear function (8×4 weight matrix) exportable to a 512-entry hardware lookup table with 8-bit quantized state space.

Phase 0 results: learned policy achieves +75% accuracy improvement over round-robin routing, within 2.8% of oracle performance (which knows the best engine for each task a priori), at 3x lower energy cost than always-parallel execution.

E. Unified Memory Fabric (UMF)

All engines share a tagged memory fabric where every value carries metadata: producing engine, confidence score, and timestamp. This enables the Orchestrator to make data-dependent routing decisions (e.g., route to SLU when NM confidence is below threshold) and supports the epistemic status system that tags every NXRI output as @proven, @probable, @speculative, or @unknown.

III. NXLang and Compiler

A. Language Design

NXLang is a domain-specific language with four block types, each mapping to an engine: perception (NM), knowledge (SLU), simulate (CSE), orchestrate (OC). The type system includes 6 primitive types (Int, Float, Bool, String, Probability, Confidence) and 8 compound types (Spike, SpikePattern, Symbol, Triple, Rule, CausalGraph, CausalNode, ActionPlan) with engine-tagged provenance. Rule syntax follows Prolog conventions (head :- body) for the knowledge blocks, while causal model declarations use arrow chains (cause → mechanism → effect).

B. Compilation Pipeline

The compiler processes NXLang source through six stages: (1) Lexer (40+ token types including block keywords, Prolog-style operators, and engine tags); (2) Recursive descent parser producing an AST with 17 node types; (3) Type checker validating predicate arity, variable binding, config ranges, and cross-block type compatibility; (4) NXIR lowering to SSA-form IR with engine annotations; (5) Optimization passes (dead code elimination, parallel region identification, cross-engine format optimization); (6) Backend code generation for Brian2 (NM), Python Datalog (SLU), NumPyro (CSE), and Python orchestration (OC).

C. NXIR Intermediate Representation

NXIR is an SSA-form intermediate representation where each instruction carries an engine annotation (@nm, @slu, @cse, @oc) and a typed result. This enables cross-engine data flow analysis, confidence-gated branching (fallback paths compiled as conditional branches on confidence values), and provides the foundation for future FPGA/ASIC synthesis. The optimizer identifies parallel regions where operations on different engines can execute concurrently, and minimizes cross-engine data format conversions.

IV. NXRI: NX Reasoning Intelligence

NXRI is a reasoning layer built on top of the three engines that provides capabilities beyond rule application. Its distinguishing feature is epistemic honesty: every output carries a calibrated status tag, and the system explicitly reports knowledge gaps rather than generating plausible-sounding but ungrounded answers.

A. Epistemic Status System

Every NXRI output is tagged with one of four epistemic statuses: @proven (logically derived from loaded rules via valid inference chains), @probable (supported by statistical evidence or causal inference with confidence ≥0.65), @speculative (abductive hypothesis with partial support, requiring verification), or @unknown (explicit knowledge gap — no matching rules or patterns found). This is the anti-hallucination mechanism: the system never presents speculation as fact, and always distinguishes between what it knows, what it infers, and what it does not know.

B. Gap Detection

When processing observations, NXRI computes knowledge coverage (fraction of observations matched to known facts), identifies partial rule matches (rules where some but not all body atoms are satisfied), and flags novel combinations (observation sets never previously encountered). This enables the system to produce outputs like: "I matched 3 of 5 symptoms to pneumonia; 2 symptoms (jaw_pain, bilateral_leg_weakness) do not appear in any loaded knowledge base."

C. Inductive Rule Learning

Given raw observational data (not rules), NXRI discovers statistical associations through three strategies: (1) single-feature co-occurrence mining with confidence and lift thresholds; (2) multi-feature combination analysis (2- and 3-way feature conjunctions); (3) differential pattern extraction identifying features that distinguish between outcome classes. Discovered rules include confidence scores, support counts, and lift values, and are explicitly marked as @speculative with their statistical evidence.

Phase 0 results: from 20 unlabeled patient records, discovers 52 rules including "chest_pain AND sweating → myocardial infarction" (confidence=1.0, support=3/3). Correctly diagnoses a novel patient case not present in the training data. Generalizes to coding (33 bug patterns from 20 reports), legal (32 case patterns from 20 disputes), and engineering (43 failure patterns from 20 incident reports) — with zero code changes to the induction engine.

D. Hypothesis Testing with Bayesian Evidence Updating

Given competing hypotheses, NXRI generates domain-specific testable predictions, each scored by discriminative power (how well the test distinguishes between hypotheses). As evidence is received, hypothesis confidence is updated via Bayesian conditioning: confirmed predictions increase confidence proportional to discriminative power; refuted predictions decrease it.

Phase 0 results: correctly identifies the top hypothesis in all 4 domains tested. In the medical domain, given troponin=elevated and ECG=st_elevation, MI confidence rises to 0.80 while pneumonia

(chest_xray=normal) and PE (d_dimer=normal) drop to 0.30. The system also suggests the most informative next test to perform.

E. Compositional Reasoning

Given typed function specifications (e.g., Python standard library signatures with input/output types and natural language behavior descriptions), NXRI discovers function compositions through backward search over a type compatibility lattice. The system discovers chains like str.split → len for "count words in a string" by recognizing that split returns a List and len accepts a Sequence (which List IS-A).

The compositional reasoning pipeline includes: (1) type-flow composition search (Levels 1–8); (2) algorithm template matching for 10 known patterns including BFS, dynamic programming, and greedy algorithms (Level 9); (3) problem decomposition into solvable sub-problems (Levels 10–11); (4) semantic verification that rejects type-compatible but semantically incorrect compositions; (5) calibrated confidence scoring. Phase 0 results: 87% accuracy across 60 problems at 12 difficulty levels (5 problems per level), with 8 honest "beyond capability" reports. We note the test set is small (n=60); expanding to hundreds of problems per level with statistical confidence intervals is a priority for Phase 1 validation.

V. Evaluation

A. Core Engine Benchmarks

Table. *Core engine benchmark results. All targets met or exceeded.*

Benchmark	Result	Target	Baseline
NM One-Shot (random, 1024n)	100%	>80%	Loihi: 93% (supervised)
NM Forgetting (8 sequential)	100% retention	>50%	CNN: catastrophic
SLU Multi-Hop (11.9K triples)	0.31ms, 100%	<10ms	Z3: 1.19ms (7x slower)
CSE Sachs Discovery	F1=0.786	>0.400	PC: 0.400, GES: 0.400
CSE Standard Networks	3/3 wins	2/3	PC algorithm baseline
Orchestrator RL Policy	+75% vs robin	>5%	Oracle: +2.8% gap
Cross-Engine Pipeline	100%	>single	Best single: 20%
ARC-AGI Real (400 puzzles)	8.5%	Honest	o3: 87.5%, GPT-4o: 5%

B. NXRI Reasoning Benchmarks

Table. *NXRI reasoning evaluation across difficulty levels and domains.*

Capability	Result	Method
Composition (L1-8)	95% (38/40)	Type-flow search over function KB
Algorithm Recognition (L9)	100% (5/5)	10 algorithm templates
Decomposition (L10-11)	60% (6/10)	Pattern + linguistic decomposition
Total Algorithmic (L1-12)	87% (52/60)	Full 5-stage pipeline

Domain Generalization	7/7 tests across 4 domains	Medical (2), coding (3), legal (2), engineering (2)
Hypothesis Testing	4/4 domains correct	Bayesian evidence updating
Inductive Rule Discovery	4/4 domains	Co-occurrence + differential mining

C. Training Efficiency

Table. *Training cost comparison. NXPU requires no gradient-based training. SLU uses loaded knowledge; NM uses single-exposure STDP; CSE uses interventional experiments.*

Method	Data Required	Cost	Energy/Pattern
NXPU NM (one-shot STDP)	1 example	~$0	7.6 µJ
NXPU SLU (knowledge load)	0 examples	$0	—
NXPU CSE (active discovery)	~200 interventions	~$0	—
CNN fine-tune (1 example)	1 example	~$0.01	~5,000 µJ
LLM fine-tune	10K+ examples	~$1K	—
LLM pre-training	13T tokens	~$100M	—

VI. Energy and Latency Projections

Energy projections are order-of-magnitude estimates based on published per-operation costs from comparable hardware: Intel Loihi spike processing at 23 pJ/synaptic operation [2], 7nm SRAM read at 5 pJ/64-bit, CAM search at approximately 50 pJ/pattern (estimated from TCAM literature), and QRNG sampling at 10 pJ/number. GPU comparison uses NVIDIA H100 at 700W TDP amortized over the full task duration, which overstates GPU energy for short tasks. These estimates carry significant uncertainty and should be treated as directional indicators of potential advantage, not precise predictions.

Table. *Order-of-magnitude energy estimates (hypothetical 7nm ASIC). Actual FPGA measurements expected 10-20x worse. GPU numbers assume full TDP amortization and likely overestimate GPU energy for short tasks.*

Benchmark	NXPU (projected)	GPU (H100 est.)	Ratio
NM One-Shot	3.0 µJ	21,000 µJ	7,000x
SLU Multi-Hop	7.3 µJ	570,000 µJ	78,000x
CSE Causal	42.2 µJ	210,000 µJ	5,000x
Cross-Engine Pipeline	9.1 µJ	196,000 µJ	21,000x

These estimates assume an ASIC implementation at 7nm that does not yet exist. The FPGA prototype (Phase 1) will be 10–20x less efficient than the ASIC target, potentially reducing the advantage to 250–4,000x over GPU — still significant if confirmed, but far from the headline figures above. The GPU energy comparison amortizes full H100 TDP (700W) over the task duration, which overestimates GPU energy for millisecond-scale tasks. A fairer comparison would measure actual GPU power draw during the specific workload. We present these numbers as motivation for hardware development, not as validated claims.

VII. Comparison to Existing Systems

Table. *Architectural comparison with existing AI accelerators.*

Feature	NXPU	Loihi 2	Graphcore	H100	Cerebras
Neural	Spiking (NM)	Spiking	Dense DNN	Dense DNN	Dense DNN
Symbolic	Native (SLU+CAM)	None	None	None	None
Causal	Native (CSE)	None	None	None	None
Routing	Learned (OC)	Fixed	Fixed	Fixed	Fixed
Learning	STDP (online)	STDP	Backprop	Backprop	Backprop
Training data	Zero	Minimal	Large	Massive	Massive
Inf. power	~1W (proj.)	~1W	~150W	~700W	~20kW

The NXPU is unique in providing native hardware support for all three computation paradigms. Intel Loihi 2 supports spiking neural networks but cannot perform symbolic reasoning or causal inference. Graphcore IPU and NVIDIA H100 accelerate dense neural network operations but provide no advantage for symbolic search or probabilistic sampling. No existing processor implements a learned routing policy between heterogeneous AI engines.

VIII. Limitations and Future Work

A. Known Limitations

- All hardware performance numbers are projections from software simulation, not measurements from fabricated silicon or FPGA
- ARC-AGI program synthesis achieves 8.5% solve rate on the ARC-AGI-1 training set, well below current SOTA (o3 at 87.5% on ARC-AGI-1 semi-private eval in high-compute configuration [9]). The limitation is DSL coverage (55 primitives), not architectural capacity. The NXPU solves each solvable puzzle in <1ms vs seconds for LLM approaches, demonstrating speed advantage on the subset it can handle
- NM sparsity averages ~50% neuron activation per inference; biological networks achieve 1-5%, suggesting room for improved inhibitory dynamics
- Level 12 compositional reasoning (genuine algorithm invention) remains unsolved and is correctly reported as "beyond capability" — an open research problem
- Cross-domain analogical reasoning requires broader concept ontologies than currently implemented

B. Phase 1: FPGA Prototype

Target platform: Xilinx XCZU7EV (Zynq UltraScale+). The initial prototype focuses on the SLU engine, implementing CAM-based triple lookup and hardware-accelerated unification. Success criterion: demonstrate ≥10x latency improvement and ≥100x energy improvement over the Python simulation on equivalent Datalog queries. Timeline: 6 months.

C. Phase 2: ASIC Tape-Out

Full three-engine chip at 7nm with silicon photonic interconnect for inter-engine communication and quantum random number generator (QRNG) for the CSE sampling pipeline.

D. Phase 3: QuantaCloud

NXPU-as-a-service platform for enterprise neurosymbolic workloads. Target domains: defense (causal threat assessment), medical (knowledge-based diagnosis with causal outcome prediction), financial (regulatory compliance with auditable reasoning), and autonomous systems (perceive-reason-plan in a single chip).

IX. Conclusion

The NXPU demonstrates that heterogeneous neurosymbolic computation is both feasible and beneficial. The combination of spiking neural networks, symbolic Datalog inference, and causal do-calculus — coordinated by a learned orchestrator — produces reasoning capabilities that no single-paradigm architecture can replicate.

The NXRI reasoning layer provides genuine reasoning with calibrated confidence, minimal supervision data, and domain-general applicability. The system discovers rules from unlabeled data, tests hypotheses with Bayesian evidence updating, composes solutions from typed specifications, and explicitly reports what it does not know. These capabilities generalize across medical, coding, legal, and engineering domains without code modification.

The architecture is ready for hardware validation. One real measurement on the FPGA prototype is worth more than any number of software benchmarks. Phase 1 will confirm whether the energy and latency projections that make NXPU a transformative platform are achievable in silicon.

References

[1] K. Sachs, O. Perez, D. Pe'er, D. A. Lauffenburger, and G. P. Nolan, "Causal protein-signaling networks derived from multiparameter single-cell data," Science, vol. 308, no. 5721, pp. 523–529, 2005.

[2] M. Davies et al., "Loihi: A neuromorphic manycore processor with on-chip learning," IEEE Micro, vol. 38, no. 1, pp. 82–99, 2018.

[3] W. Maass, "Networks of spiking neurons: The third generation of neural network models," Neural Networks, vol. 10, no. 9, pp. 1659–1671, 1997.

[4] J. Pearl, Causality: Models, Reasoning, and Inference, 2nd ed. Cambridge University Press, 2009.

[5] L. de Moura and N. Bjorner, "Z3: An efficient SMT solver," in Proc. TACAS, Springer LNCS 4963, pp. 337–340, 2008.

[6] F. Chollet, "On the measure of intelligence," arXiv preprint arXiv:1911.01547, 2019.

[7] X. Zheng, B. Aragam, P. K. Ravikumar, and E. P. Xing, "DAGs with NO TEARS: Continuous optimization for structure learning," in Advances in Neural Information Processing Systems, vol. 31, 2018.

[8] J. Kirkpatrick et al., "Overcoming catastrophic forgetting in neural networks," Proc. National Academy of Sciences, vol. 114, no. 13, pp. 3521–3526, 2017.

[9] OpenAI, "o3 and o4-mini system card," OpenAI Technical Report, Apr. 2025. o3 scored 87.5% on ARC-AGI-1 (semi-private eval, high-compute configuration); o3-mini scored 76.7% (low-compute). Our 8.5% comparison uses the ARC-AGI-1 public training set, the same task distribution.

[10] S. Abiteboul and V. Vianu, "Generic computation and its complexity," in Proc. 23rd ACM STOC, 1991, pp. 209–219.